|  |
| --- |
| CSCE 401: Software Engineering |
| Saving Face |
| Facial Recognition using the Intel Creative Camera |
|  |
| Andrew Mason ([aemason@alaska.edu](mailto:aemason@alaska.edu))  Keith Schneider ([1900Geek@gmail.com](mailto:1900Geek@gmail.com))  Jacob Dempsey ([Jacob@JacobsChaos.com](mailto:Jacob@JacobsChaos.com)) |
| **12/2/2013** |

|  |
| --- |
| In our project, Saving Face, the Intel Creative Camera’s depth sensor is used to collect data points from which a 3D model is produced, and used for future comparison to identify users. This report covers the development process, the algorithms used, and code implementation. |

Contents

[Overview of Requirements and Features: 1](#_Toc373820763)

[Intel Creative Camera Specifications 1](#_Toc373820764)

[Recommended System Configuration 1](#_Toc373820765)

[Feature List: 2](#_Toc373820766)

[Development Estimates and Actual Progress: 3](#_Toc373820767)

[Description of the Design: 4](#_Toc373820768)

[Description of the Finished Product: 5](#_Toc373820769)

[Description of Tests: 5](#_Toc373820770)

[Description of the Process: 5](#_Toc373820771)

[Who Worked on What: 5](#_Toc373820772)

[References 7](#_Toc373820773)

# Overview of Requirements and Features:

Our project requires the use of an Intel Creative Camera (*Figure 1*) for the collection of three dimensional data points used in all of the functionality of our project.

Figure : Intel Creative Camera.

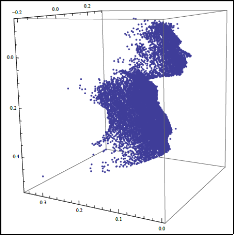
Saving Face features 3D model reconstruction and comparison of a person’s face (Figure 2). The comparison is performed by mapping retrieved depth values to a 3D model based off of the yaw, pitch, and roll of the face centering the tip of the user’s nose to the origin of the 3D coordinate model (Figure 3). The model data is collected over a predefined number of frames to give a better estimate of the user’s face in the model. The user’s initial model, or reference model, is stored in a database, and the identification models are then compared to the models in the database to find the closest match.

Figure : 3D model data.

## Intel Creative Camera Specifications

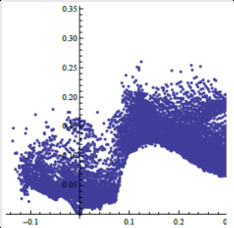
Intel® Developer Kit (2013) 

Figure : data centered at the origin.

Maximum RGB Video Resolution: HD 720p (1280x720)

Maximum IR Depth Resolution: QVGA (320x240)

Frame Rate: 30 fps

FOV (Diagonal): 73°

Range: 0.5ft to 3.25ft

Size: 4.27”x2.03”x2.11”

Weight: 9.56 oz

Power: Single USB 2.0 (power < 2.5W)

Dual-array microphones

RGB + Depth frame sync

## Recommended System Configuration

Intel® Developer Kit (2013)

PC with 2nd Generation or newer Intel® Core™ processor

Windows® 7 with Service Pack 1 or higher

4 GB system memory

USB 2.0 port

## Feature List:

|  |  |  |
| --- | --- | --- |
| **Story Name** | **Description** | **NUTS** |
| Fixed Point Orientation | Find X Y Z of at least three fixed points; Or 1 and Yaw Pitch and Roll | 2 |
| Transform Module | A Module that applies the Transform to bring the points into a common plane | 6 |
| Build Model Module | A Module that takes the transformed data and makes a model representation | 5 |
| Compare Module | A Module that takes a stream and compares it to a list of Models and returns a ranking | 5 |
| Load Module | Load a model from a file | 2 |
| Save Module | Saves a model to a file | 2 |
| Debug Terminal | Displays Helpful Information | 2 |
| Save Depth and Color Video | A Module that saves video for later re-use | 4 |
| Display Streams | A Module that displays Depth and Color Streams | 1 |
| User Information Input | A GUI Interface for inputting model Information | 3 |
| Save Image Data | A Module To Save a User Photo for Later Display | 2 |
| Load Color/Depth Video | A Module To Load Video Playback for Algorithm Refinement and Debugging | 4 |
| Load user Image | A Module to Load and display a user image | 2 |
| Activity: Model Collection | Collect Model Data To Test | 3 |

# Development Estimates and Actual Progress:

For most sections of our project, the estimates were relatively accurate. Toward the end, some were altered by a few points. Our burn up chart (*Figure 4*) accurately represents the team’s completion of nuts during this process. Our project had iterations that were not truly defined by iterations of equal amounts of work time so it is hard to encapsulate the iterations in the chart. Our first iteration may have had 10 – 15 actual hours of work, where as our last iteration easily had over 80 hours of work. From the time that the project started to now we noticed an exponential increase of hours put in vs. time passed. The burn up chart increases as we noticed we had to revisit a previous user story, and we switched from a debug terminal to a GUI. There was also a function that was not implemented in the PC SDK that we were relying on. In realization that this method was not implemented we had to create our own function to convert color coordinates to depth coordinates.

Our estimates mid semester showed us that we were not going to finish the project on time. However thanks to complete devotion at this critical point we tripled the amount of work being done. The estimates taken several weeks after showed us that we could finish the project on time.

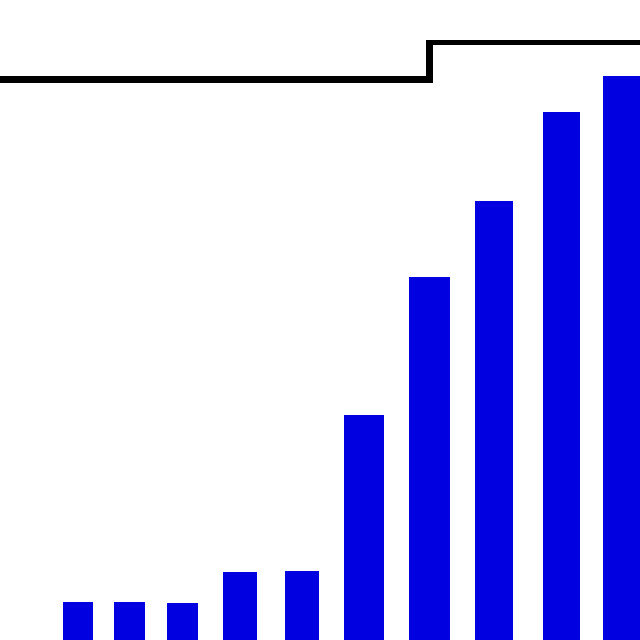


Figure : Saving Face burn up chart.

# Description of the Design:

Keith Schneider had some great foresight into the project. He designed early prototypes of the algorithms we needed to align the depth images for all models into a byte array. The functions used output from the Intel Creative Camera that was loaded into Mathematica for testing. Here is some pseudo code of the model building algorithm:

Generic Algorithm for constructing the model

While[Frames < N]

Read in byte stream

Get Fixed Point, and Yaw Pitch and Roll From SDK

Calculate Transformation Matrix Based On Yaw Pitch and Roll

Determine Relative Location Of Head

For[Vertices]

Apply Linear Transform to bring point relative to origin

Apply Rotational Transform To Align Head

Map Point to Model Byte[]

Increase that point by one

Save Model

End

In order to compare models it's essential that all of the models line up the same in our vertices array. Using the yaw, pitch, and roll we could calculate a transformation matrix that would be applied to all vertices to make all models the same rotation. We also centered the nose of each model at the origin of the 3D plane. So now our models are not only in the same rotation but also in the same position.

Add more about compare method.

# Description of the Finished Product:

This should be more wordy with the specifics of our completed project.

Saving Face is an application designed to build and compare 3D models using the Intel Creative Camera. Saving Face takes a depth image of a users face and builds a 3D model of it, then saves the model into the database. On a face comparison it will iterate through all of the database models comparing vertices with each other to determine a match.

# Description of Tests:

Add anything you want here, there was more to testing than just this

We used an automated testing system for the development process, we did do testing first but sometimes we did testing after. The automated testing system was an excellent way to track bugs and fix them. Testing first gave us more insight on how to avoid bugs in a given function. Testing first may feign a delay of progress but it saved so much time in the long run. We have multiple tests for functions and did a good job at determining a function was working based on an correct output with given inputs.

# Description of the Process:

At the beginning of development we did good at staying true to the agile method. We created user stories and had client meetings that would help us get started. Later on in the development process we deviated from the agile methodology. When we realized we had to play a bit of catch up, we started tackling user stories independently without acceptance tests. With time constraint at our backs we had to make the biggest jump we could in a short amount of time.

# Who Worked on What:

This may need to be more of a task completion rather than titles.

Keith Schneider:

Project analyst, project lead, concept designer, lead programmer and

lead tester.

Andrew Mason:

Assistant programmer, assistant tester, and assistant designer.

Jacob Dempsey:

Assistant programmer and lead GUI designer.

Overall Experience:

I think each of us should throw some of our input into this section of the report.

One of the biggest challenges was creating a constant schedule for our project. While other classes interfered we found that our hours of work varied week by week. Creating a definitive schedule of what needed to be done and how many hours should be done in an iteration for each team member would have helped a lot. We learned many new programming techniques and also a new and wonderful SDK. This was a very large project and in hind sight we believe that more team meetings to get the team organized would have been be better. Allowing more direction for individual tasks and also adding a better understanding of the code implementation.

# References

Intel Corporation. (2013) “Creative Intel Developer Kit”. Retrieved from http://download-software.intel.com/sites/default/files/article/325946/creativelabs-camera-productbrief-final.pdf December 2, 2013.